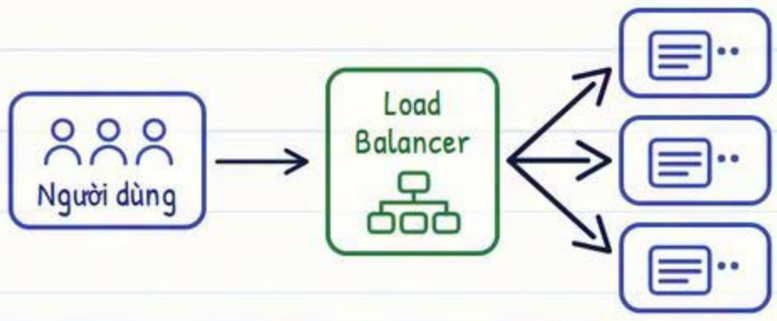
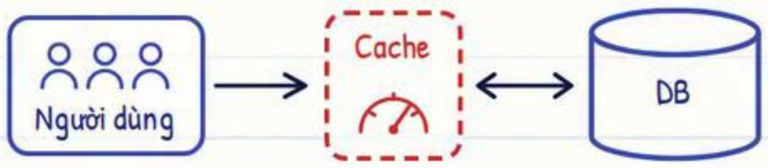
Các khái niệm System Design

1. Load Balancer



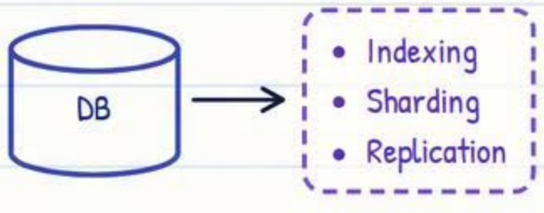
Trường hợp sử dụng:
- Ứng dụng web
- API
- Microservices

2. Cache



Trường hợp sử dụng:
- Lưu session
- Phản hồi API
- Bảng xếp hạng

3. Database



Trường hợp sử dụng:
- Dữ liệu người dùng
- Giao dịch
- Phân tích dữ liệu

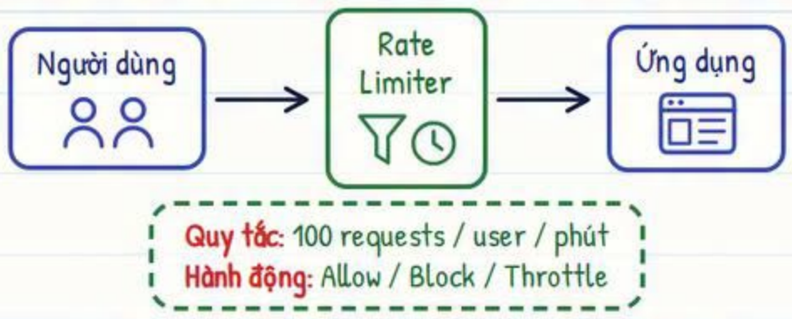
4. Microservices

Trường hợp sử dụng:
- Ứng dụng TMĐT
- Hệ thống ngân hàng
- Nền tảng SaaS

5. Message Queue

Trường hợp sử dụng:
- Xử lý đơn hàng
- Gửi email
- Hệ thống event-driven

6. Rate Limiter



Trường hợp sử dụng:
- API
- Lượt đăng nhập
- Yêu cầu thanh toán

7. Search Service

Trường hợp sử dụng:
- Tìm kiếm trên sàn TMĐT
- Tìm kiếm tài liệu
- Tìm kiếm log

8. File Storage

Trường hợp sử dụng:
- File media
- File người dùng tải lên
- Static assets
- Sao lưu & lưu trữ

9. Notification System

Trường hợp sử dụng:
- Cảnh báo
- Nhắc nhở
- Cập nhật
- Khuyến mãi

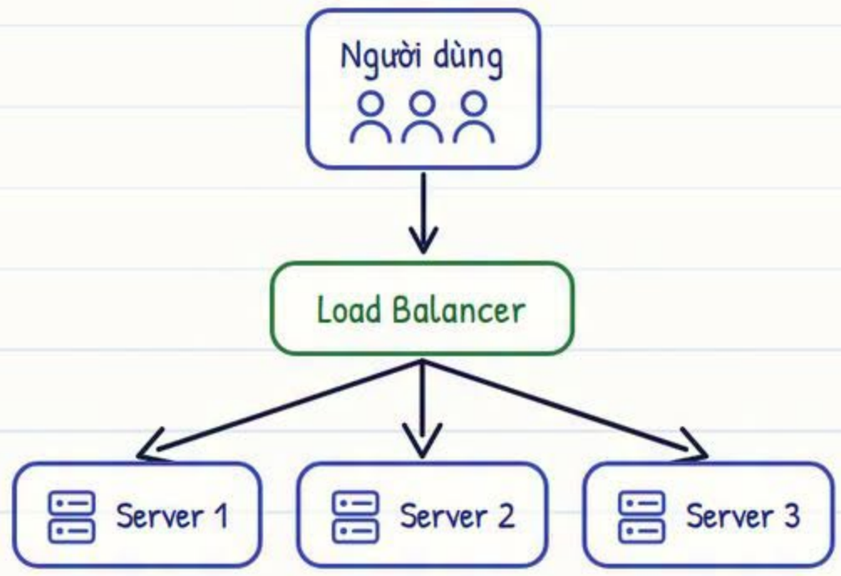
10. Real-time Analytics

Trường hợp sử dụng:
- Dashboard
- Phát hiện gian lận
- Giám sát trực tiếp
- Phân tích hành vi người dùng

Các khái niệm System Design

1. Load Balancer

- Phân phối traffic đến nhiều server khác nhau.
- Ngăn không để bất kỳ server nào bị quá tải.
- Tăng tính sẵn sàng và khả năng chịu lỗi của hệ thống.



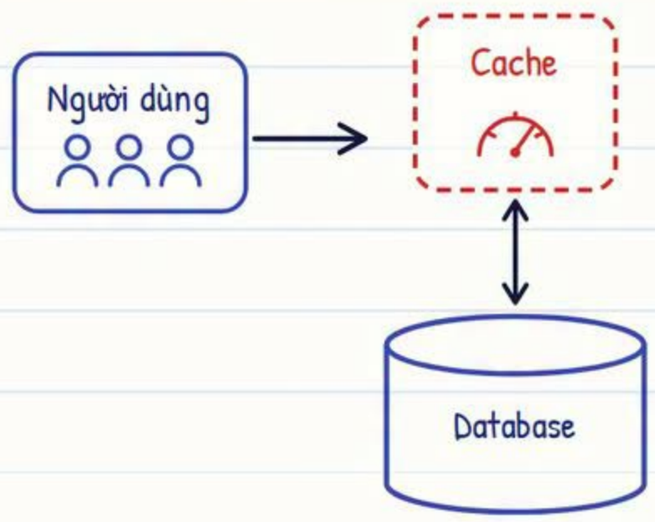
Trường hợp sử dụng:

- Ứng dụng web
- API
- Microservices

★ Đóng vai trò là điểm vào duy nhất, đảm bảo traffic được định tuyến hiệu quả tới các server đang hoạt động tốt.

2. Cache

- Lưu dữ liệu được truy cập thường xuyên trong bộ nhớ.
- Giảm tải cho database và cải thiện thời gian phản hồi.
- Dữ liệu trong cache chỉ là tạm thời nhưng rất nhanh.



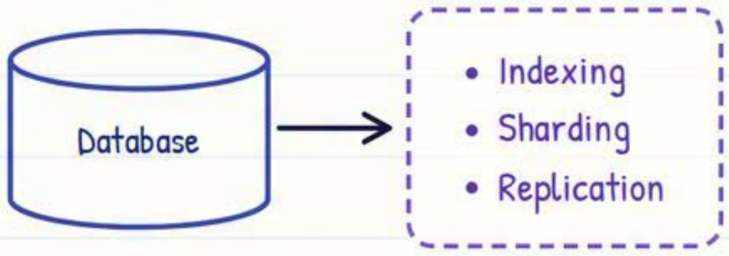
Trường hợp sử dụng:

- Lưu session
- Phản hồi API
- Bảng xếp hạng
- Danh mục sản phẩm

★ Cache nằm giữa người dùng và database để trả dữ liệu thật nhanh và giảm độ trễ.

3. Database

- Lưu trữ, quản lý và tổ chức dữ liệu lâu dài.
- Chọn loại phù hợp với nhu cầu (SQL / NoSQL).
- Khái niệm quan trọng:
  Indexing, Sharding, Replication.



Trường hợp sử dụng:

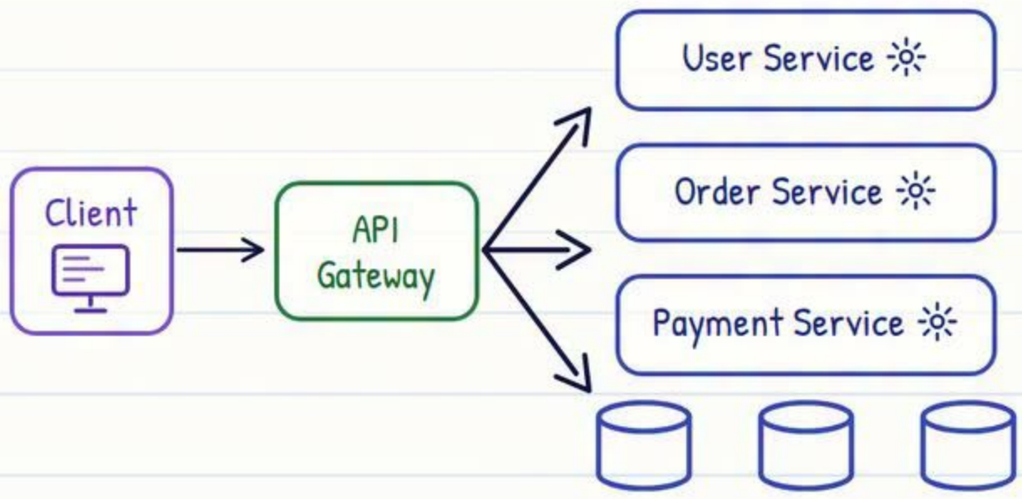
- Dữ liệu người dùng
- Giao dịch
- Phân tích dữ liệu

★ Database là xương sống cho việc lưu trữ lâu dài, đảm bảo tính nhất quán và độ tin cậy của dữ liệu.

Các khái niệm System Design

4. Microservices

- Chia một ứng dụng lớn thành nhiều service nhỏ, độc lập.
- Mỗi service có trách nhiệm và database riêng.
- Tăng khả năng mở rộng, tính linh hoạt và tốc độ triển khai.



Trường hợp sử dụng:

- Ứng dụng TMĐT
- Hệ thống ngân hàng
- Nền tảng SaaS

★ Microservices giúp các team phát triển, triển khai và mở rộng service một cách độc lập, tạo ra hệ thống linh hoạt hơn.

5. Message Queue

- Cho phép giao tiếp bất đồng bộ giữa các service.
- Message được lưu trong queue cho tới khi được xử lý.
- Giảm phụ thuộc và tăng độ tin cậy cho hệ thống.

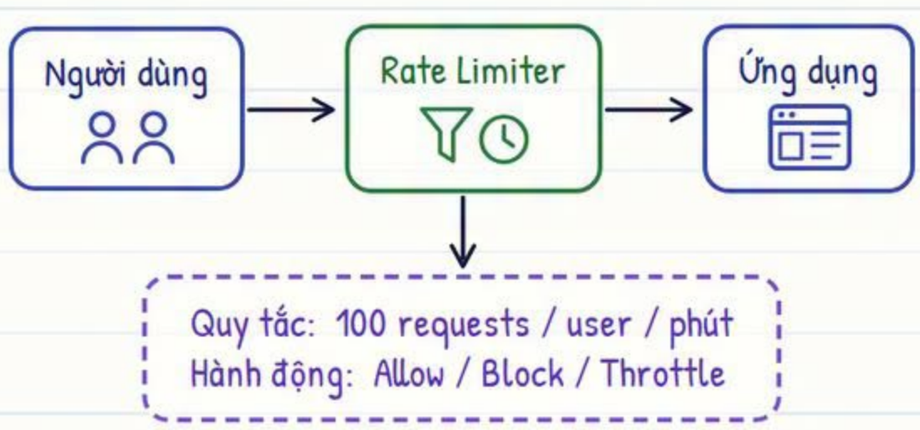
Trường hợp sử dụng:

- Xử lý đơn hàng
- Gửi email
- Hệ thống event-driven

★ Message queue đóng vai trò bộ đệm giữa các service, đảm bảo dữ liệu không bị mất kể cả khi một service tạm thời ngừng hoạt động.

6. Rate Limiter

- Giới hạn số lượng request một user gửi trong một khoảng thời gian.
- Ngăn lạm dụng và bảo vệ hệ thống khỏi quá tải.
- Đảm bảo tài nguyên được sử dụng công bằng.



Trường hợp sử dụng:

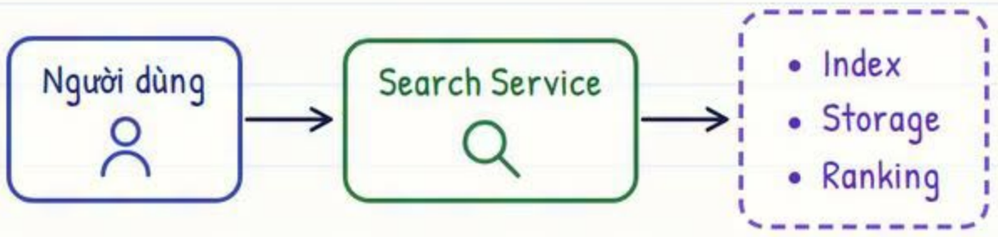
- API
- Lượt đăng nhập
- Yêu cầu thanh toán

★ Rate limiting giúp giữ hệ thống ổn định và mang lại trải nghiệm tốt hơn cho mọi người dùng.

Các khái niệm System Design

7. Search Service

- Trả về kết quả tìm kiếm nhanh và phù hợp.
- Sử dụng indexing và phân tích văn bản.
- Xử lý hiệu quả hàng triệu truy vấn mỗi ngày.



Trường hợp sử dụng:

- Tìm kiếm trên sàn TMĐT
- Tìm kiếm tài liệu
- Tìm kiếm log

★ Search service giúp người dùng nhanh chóng tìm được thứ họ cần với kết quả phù hợp.

8. File Storage

- Lưu trữ và phân phối các file lớn như hình ảnh, video, tài liệu.
- Giảm tải file khỏi các app server.
- Cải thiện hiệu năng và khả năng mở rộng.

Trường hợp sử dụng:

- File media
- File người dùng tải lên
- Static assets
- Sao lưu & lưu trữ

★ Dịch vụ file storage cung cấp nơi lưu trữ bền vững, dễ mở rộng và tiết kiệm chi phí cho mọi loại file.

9. Notification System

- Gửi thông báo real-time qua nhiều kênh khác nhau.
- Tách logic thông báo ra khỏi ứng dụng chính.
- Đảm bảo gửi đúng lúc và tăng tương tác của người dùng.

Trường hợp sử dụng:

- Cảnh báo
- Nhắc nhở
- Cập nhật
- Khuyến mãi

★ Hệ thống notification giúp người dùng luôn được cập nhật, qua đúng kênh vào đúng thời điểm.

10. Real-time Analytics

- Xử lý dữ liệu real-time và tạo ra insight.
- Hoạt động trên dữ liệu streaming.
- Hỗ trợ việc ra quyết định và giám sát hệ thống.

Trường hợp sử dụng:

- Dashboard
- Phát hiện gian lận
- Giám sát trực tiếp
- Phân tích hành vi người dùng

★ Real-time analytics biến dữ liệu trực tiếp thành insight hữu ích để ra quyết định nhanh và thông minh hơn.

Các thành phần này kết hợp lại giúp chúng ta thiết kế hệ thống dễ mở rộng, đáng tin cậy và hiệu quả!